IBM 3270 User Guide (IDSH)

I.P. Sharp Associates

I. P. Sharp Associates Limited
2 First Canadian Place, Suite 1900
Toronto, Ontario, Canada
M5X 1E3

Telephone 416 364-5361
Telex 0622259 I P SHARP TOR

(c) 1984 by I. P. Sharp Associates Limited.

All rights reserved. This book or parts thereof may not be reproduced by any means without the written permission of I. P. Sharp Associates Limited.

Printed in Canada
December 1984
Publication Code: 0346 8412 E2

IBM 3270 User Guide (IDSH)
1 December 1984

Abstract: SHARP APL under MVS supports the IBM 3270 display stations through an auxiliary processor called IDSH (Information Display Station Handler). This document describes how the IDSH Session Manager supports the IBM3270 display station as an APL terminal and how full screen management is provided by AP124, AP126, and the arbitrary input/output interface. AP6, the session manager command processor, is also described.

Introduction, 5

A Quick Guide to the 3270 Session Manager, 7
Getting Connected to the Session Manager, 7
The Session Manager Screen, 7
Signing-on To APL, 7
Conversing with APL, 7
Vertical Scrolling, Paging and Searching, 8
Horizontal Scrolling, 9
)SESM System Command, 9
Program Function Keys, 9
Editing the Input Line to APL, 10
Some Other Commands, 11
Disconnection From the Session Manager, 11

The SHARP APL Session Manager Reference Guide, 13
Introduction, 13
Summary of Facilities, 13
Mode of APL, 15
The Next Input Line, 15
The Session Log, 15
    The Referenced Line, 16
    The Referenced Column, 16
    A Display Page, 16
    The Line Display Mode, 16
    The Referenced Line Offset, 16
Session Manager Commands, 17
Program Function Keys, 17
)SESM Pseudo-system Command, 18
General Session Manager Commands, 18
The 3270 Screen, 23
    The Session Manager Row, 23
    The APL Screen, 24
Preparation and Submission of Next Input Line, 25
Non-Data Entry Keys, 26
Updating the APL Screen, 27
The *ROLL* Setting, 28
The *AUTOPAGE* Setting, 28
The Overstrike Escape Character, 28
Input Line Overlaying, 29
Colour/Highlight Attributes, 29
3270 Session Manager Commands, 29
The Session Manager Command Auxiliary Processor (AP6), 31
Miscellaneous Messages, 33

Full Screen Management, 35

AP124 Full Screen Manager, 37

Operation Code Details, 41

1) Operations To Define Fields, 41
2) Operations to Write and Read, 46
3) Miscellaneous Operations, 48

PUBLIC FULLSCREEN WORKSPACES, 51
Workspace 5 *IBM*3270, 51
Workspace 5 *AP*124, 53
Workspace 5 *AP*124*E*, 53

*AP*126 *FULL SCREEN MANAGER*, 55
AP126 Service Requests, 58
AP126 CTL Return and Reason Codes, 60

ARBIN Full Screen Manager, 61
Constructing a Data Stream, 61
Sharp Order Codes - Detailed Description, 66
Transmitting a Complete Header and Data Stream, 70

SHARP APL under MVS supports the IBM 3270 display stations through an auxiliary processor called IDSH (Information Display Station Handler). IDSH uses the S-task facility to interface with SHARP APL and uses VTAM to support the IBM 3270's. When IDSH connects a 3270, it determines what features the device has and whether or not it has the APL character set.

The IBM 3270 device operates in two modes: standard screen and full screen. In standard screen mode, the device operates as an APL terminal with system defined characteristics. While in full screen mode, the screen is under the complete control of an APL application program.

This document provides a full description of how the device operates in both modes. Included is a description of workspace 5 *IBM*3270 which contains functions that provide full screen management support using AP124.

For an in-depth description of 3270s, see the IBM manual GA27-2749, IBM 3270 **Information Display System Component Description**.

NOTE: Origin 0 is used throughout this document.

The SHARP APL 3270 session manager enables a user to run an APL session from a 3270 terminal in a way that somewhat resembles a "line-by-line" scrolling terminal. In addition, a "log" of the session-to-date is kept so that browsing back through the session is possible.

The following is a quick guide to how to use the session manager, intended to help a user become minimally aquainted with the facilities. A detailed account of the facilities is given in the next section.

Getting Connected to the Session Manager

The procedure to gain an initial connection to the session manager will vary from one installation to another. Once this is achieved, the session manager will attempt to determine the exact characteristics of the terminal. If it is unable to do so, a "menu" of possible terminal types will be presented. You will have to select (by number) the type that most closely describes your terminal.

The Session Manager Screen

The initial session manager screen will then be presented. The top row will appear as follows:

U INP *LINE* 0 *COL WRAP*

indicating that the session manager is expecting input from you (*U*, for **unlocked**), APL is expecting input (*INP*), we are currently on line 0 of the session log, and lines of the session will be displayed **wrapped** (i.e. displayed on more than one row of the screen if necessary). The remainder of the screen will display the first part of the session log (containing a banner-like logo), and the cursor will be placed on the first unused row of the screen, indicating where you are expected to type (the **APL input area**).

Signing-on To APL

Enter your APL user number and password in the usual way:

```
)1234567:lock
```

and press ENTER. Anything typed here will not be displayed, to protect your signon password. The normal APL signon messages will be displayed on the screen immediately below where you typed.

Conversing With APL

You can now proceed through a normal APL "conversation" in the same way: enter input where the cursor indicates (after the last prompt from APL), and output will be displayed following your input. When the screen fills up without displaying all available output, the left end of the top screen row will appear as follows:

U H

The *H* indicates that the session manager is **holding** output.

To display this held output, press ENTER. The held output will be displayed starting a few rows down from the top of the screen. The first such line becomes the "referenced line", its number appears in the first screen row after the *LINE* heading, and the line itself is usually highlighted in some way on the screen (e.g. painted a distinguishing colour).

When APL is not ready for you to type, the left part of the top screen row will appear as follows:

L RUN

The *L* indicates that the screen is **locked**. To send an "attention" signal to APL in this state, press PA1.

If APL is ready for input, and you wish to send an input interrupt to APL, press PA1 also.

Vertical Scrolling, Paging and Searching

To "browse" through the session log, you can use a number of session manager **commands**. For example:

To scroll backward 5 lines type:

```
-5
```

after the *LINE* heading on the top screen row, and then press ENTER. The **referenced line** will decrease by 5.

To scroll forward 14 lines, type:

```
+14
```

To force the referenced line to be line 23, type:

```
23
```

in the same field.

To "page" backward exactly 1 screen display type:

PAGE-1

in the field following the *U* *INP* status display (the **command area**), and then press ENTER.

To page forward, type *PAGE+1* in the same field.

To search backwards from the referenced line for the string *)LOAD* type *BACK ')LOAD'* in the command area. The referenced line will become the first preceding line that contains the string (if such a line exists).

To search forward, type *FORW ')LOAD'* in the same area.

If, as a result of scrolling or paging, the area for typing APL input is no longer on the screen, press ENTER (with no other changes to the screen) to revert to an "end of log" display that does contain that area.

Horizontal Scrolling

Normally, the lines of the log are displayed **wrapped**, meaning that long lines occupy several screen rows if necessary. It is possible to change the mode of display to **clipped**, in which only as much of each line as will fit on one screen row will be displayed. In this mode, it is also possible to control the leftmost column of the displayed lines, and by changing this column, you can perform horizontal scrolling of the display. To change the mode to "clipped", type:

```
0
```

following the *COL* heading on the first screen row.

To scroll to the right 5 columns, type +5 in the same field. To scroll to the left 3 columns, type -3. To make column 24 the leftmost displayed column, type 24. To change the display mode back to "wrapped", type *W*.

)SESM System Command

Any of the commands can also be issued by using the system command *)SESM* as if it were an APL system command. For example, in the area where you normally type APL input, type:

```
)SESM LINE+1
```

To perform 1 line "scroll up". (The command is never actually seen by APL, but is intercepted by the session manager.)

Program Function Keys

Each of the program function keys can be "assigned" a command (or a sequence of commands) to provide a convenient way of issuing commands you use frequently. To display the current assignments of the keys, type:

```
PFK
```

in the command area. The values of any keys that have non-empty assignments will be displayed on the screen (after an "echoing" of the command in the form *)SESM PFK*).

To assign a command (say, *LINE*+1) to PF key 3, issue the command:

```
PFK 3,'LINE+1'
```

Then to perform a 1 line "scroll up", you can simply press PF key 3.

Editing The Input Line to APL

There are a number of ways of preparing, and amending, the line which is to be submitted to APL as its next line of input. Obviously, simply typing the line into the APL input area is one way. However, you can also:

- "bring down" a previous line in the log: place the cursor anywhere on the desired line, and press ENTER. The line will replace whatever is currently in the APL input area.
- "bring down" an amended version of a previous line in the log: make the amendment to the displayed version of the line by typing over it (or using the insertion and deletion facilities of the terminal).

If you use a combination of the above devices, the input line will be replaced with the catenation of the lines.

Automatic insertion: assign a PF key (say, PF key 6) with an *INS* command:

```
PFK 6,'INS C,'')LOAD 666 BOX'''
```

Then, with the cursor in the input area, press PF key 6. The string *)LOAD* 666 *BOX* will be inserted where the cursor indicates. If you also wish the line to be submitted to APL immediately, assign the PF key as follows:

```
PFK 6,'/INS C,'')LOAD 666 BOX''/ENTRY F'
```

(This assigns a sequence of 2 commands: the second specifies that entry to APL is to be **forced**).

Normally, the submission to APL of the input line you have prepared will take place when you position the cursor somewhere on the line and press ENTER (without, at the same time, "moving" (e.g. scrolling) the screen). You can avoid submitting the line to APL, therefore, by placing the cursor on the first screen row (e.g. press ALT BACKTAB), and pressing ENTER. The input line will not be submitted, and the cursor will be placed at the end of the line.

Some Other Commands

HELP - to display a list of commands.

HELP LINE - to remind you of the syntax of the *LINE* command.

DEL C - to delete lines from the log from the cursor to the end of the log.

There are also commands to control the screen overlap of lines (*ROLL*), whether to pause when the screen becomes full (*AUTOPG*), and to control the colour/highlight attributes of the screen fields (*FIELD*).

Disconnection From The Session Manager

To disconnect yourself from the session manager, sign off the APL task as normal (*)OFF*); then press PA1 or type *)OFF* in the APL input area.

If you sign off the APL task but do nothing more, you will be disconnected automatically after a few minutes.

Introduction

The APL interpreter can be considered from an "outside" perspective to be a transaction processor: in response to a line of input from the user, APL produces one or more lines of output. Taking this sequence of lines of input and output as the manifestation of a "session", the **SHARP APL Session Manager** is a facility, operating outside APL, which manages such a session in the following sense:

- the next line of APL input is maintained in such a way that it can be submitted to APL when the user is ready to do so;
- a more or less permanent log of the session's input and output lines is kept (as a sequence of lines) which can be processed to some extent by the user; for example, browsing through, or retrieving, information in the log.

The session manager can be considered to be a facility operating at the "gateway" to (but outside) APL.

The 3270 **terminal handler** is a facility that enables a 3270 terminal to be used to run an APL session. The user of the session gains access to APL through the facilities of the session manager; for example,

- parts of the session log can be displayed on the terminal screen;
- the next input line can be displayed on the screen for possible amendment, and submission to APL.

Summary of Facilities

The SHARP APL session manager provides the following facilities:

- **next input line:** the next line of input for eventual submission to APL is maintained in such a way that it can be changed repeatedly prior to its submission.
- **the session log:** when a line is submitted to, or received from, APL, it is appended to the session log. It is possible to submit lines to, and receive lines from, the session manager itself (i.e. such lines are never seen by APL). These are also appended to the session log.
- **Session Manager Commands:** a command language is available, by which the user may explictly obtain some of the services of the session manager.
- **Program Function Keys:** certain command sequences can be associated with numbered "program function keys", so that they may be conveniently invoked.

- *)SESM* **Command:** this pseudo-APL system command can be used as one of several ways of entering session manager commands.
- **Session Manager Command Auxiliary Processor (AP6):** an auxiliary processor is available for submitting session manager commands under the control of an APL program.

The 3270 terminal handler provides the following facilities:

- **Screen Display:** selected (contiguous) parts of the session log, and the next input line can be displayed on the screen.
- **Input line amendments:** the next input line can be altered by amending the line as displayed on the screen; by amending a line or lines of the session log as displayed on the screen; or simply by placing the cursor on such a displayed line.
- **Sending input to APL:** the user can control when the next line of input is sent to APL, by positioning the cursor on the screen.
- **Display of session status:** some aspects of the status of a session are dispayed on the screen (e.g. whether APL is in "run" or "input" mode).
- **Session manager row:** an area of the screen is available as one of several ways of submitting session manager commands.
- **3270 terminal commands:** some special session manager commands are available which pertain to the terminal itself (e.g. a command to control the colour/highlighting of areas of the screen).
- **Data and non-data keys:** actions are associated with the various keys: PA1, PA2, CLEAR, etc.
- *□ARBOUT/□ARBIN* **interpretation:** an interpretation of APL *ARBOUT* strings is provided which is meaningful especially (and only) for 3270 terminals. This consists primarily of the ability to write and read 3270 datastreams.
- **Fullscreen auxiliary processors (AP124/AP126):** two auxiliary processors are available for writing fullscreen applications.

It is useful to distinguish the session manager from the terminal handler, but in what follows, the combination of the two will simply be called the **session manager.**

There follows a more detailed account of the facilities and terms summarized above.

Mode Of APL

The **mode of APL** at any time is either:

- **input** mode: APL has sent the user a prompt and is expecting input from the user, or
- **run** mode: APL has not yet produced the next user prompt.

The Next Input Line

The **next line of APL input** may be changed any number of times prior to its submission to APL, and may be changed whether APL is in "run" mode or "input" mode. (See below for details on ways of changing the line).

The line is submitted to APL when:

- APL is in input mode;
- the preparation of the line is considered to be complete (see below for details)

After submission to APL, the next input line becomes an empty line. When APL changes from run mode to input mode, the last line of output (the **prompt**) is catenated to the front of the next input line.

The Session Log

When a line is submitted to, or received from APL, it is appended to the session log. This log is permanently available until the user disconnects (or is disconnected) from the session. A line of output from APL is considered complete when a carriage return or linefeed character is received. These characters are not included in the line, and a received linefeed character causes an appropriate number of blanks to be placed at the beginning of the next line. Any backspace characters are resolved by moving back in the "line so far" so that subsequent characters will **replace** those already received. The first 2 characters of *⎕AV* are suppressed.

As described later, it is possible to submit lines to, and receive lines from, the session manager itself (i.e. such lines are never seen by APL). These are also appended to the session log.

The lines in the log are stored in such a way that it is possible to determine whether the line was a line to or from APL, or to or from the session manager. In the case of lines submitted to APL, the length of any false prompt (a prompt which was produced with the use of *⎕ARBOUT* '') is also kept.

The lines in the log are numbered sequentially from 0. Since there is a physical limit on the size of the session log, when a line is appended to the log, room will

be made for it if necessary by deleting lines from the beginning of the log. The lines are not renumbered when this is done, however.

A number of quantities are associated with the session log which are useful in dealing with it:

The Referenced Line:

At any time, one of the lines of the session log (or the next input line) is designated as the **referenced line**. It can be thought of as "the line we are currently looking at". It can be changed in a number of ways, including explicit session manager commands, and provides the basic means of "browsing" through the session log.

The Referenced Column:

Similarly to the referenced line, one of the columns is designated as the currently **referenced column**. It can be changed by session manager commands, and provides a means for horizontal browsing. The leftmost column is numbered as column 0.

A Display Page:

For display purposes, lines from the session log (and the next input line) can be arranged on a rectangular page. Normally, a long line will be wrapped onto as many page rows as are needed to display the whole line.

The **page width** and the **page depth** are the number of columns and rows, respectively, of the display page.

The Line Display Mode:

For display purposes there are two **line display modes**:

- **wrapped**: in which a line of the session log will be displayed in full, occupying as many page rows as necessary;

- **clipped**: in which only as much of a line as will fit on one page row will be displayed. In this case, the first character of the line displayed will be that specified by the referenced column. In this mode, it is possible to perform horizontal browsing.

The Referenced Line Offset:

The **referenced line offset** is the number of page rows which will normally precede the referenced line when displayed. When >0, this provides a way of displaying some of the preceding context of the referenced line.

Session Manager Commands

Some of the facilities of the session manager can be explicitly requested by submitting session manager commands. There are a number of mechanisms for submitting such commands:

- *)SESM* pseudo-APL system command;
- the session manager row on the 3270 screen;
- defining and invoking program function keys;
- the session manager command auxiliary processor (AP6).

(See below for details on these various devices.)

The general form of a command is:

commandword argumentlist

It is possible to submit a sequence of commands by making the first character a non-alphabetic character and using this as the command separator. For example:

/LINE I0/ENTRY F is a sequence of 2 commands

The execution of a command results in:

- A non-negative integer completion code. 0 indicates "successful" completion of the command; the non-zero "failure" codes are listed in the section titled "Miscellaneous Messages". These non-zero codes are only of significance when the command was issued by means of the session manager command auxiliary processor (AP6).

- Zero or more character vector results. These results are manifested in ways which depend on how the command was issued. For instance, if the command was issued from the session manager row on the 3270 screen, the result(s) may appear in the command area of that row, or be appended to the session log.

The commands in a command sequence will be executed one after the other up to and including the first one which results in a non-zero completion code.

Program Function Keys

Up to 24 **program function keys**, numbered 1 to 24, can be defined. Each such key can be defined to be a session manager command or command sequence, and provides a convenient way to invoke commonly used commands. The keys can be defined by means of the session manager command *PFK*, and they can be invoked by means of the command *XPFK*, or by pressing a 3270 physical program function key.

)SESM Pseudo-system Command

The system command *)SESM* is provided as one of the means of submitting session manager commands. It can be used in any circumstance where an APL system command could be used (i.e. immediate execution), but it is intercepted by the session manager and is never seen by APL. Its argument can be any session manager command or command sequence, and its result(s) are appended to the session log. For example:

```
     )SESM PFK 1
PFK 1,'PAGE-1'
```

General Session Manager Commands

The general session manager commands are:

LINE: To alter the referenced line (and referenced line offset), or to query the number of the referenced line.

LINE [(+|-) **integer**
| **lineno**[,**integer**]]

LINE
- the result is the number of the currently referenced line.

LINE **lineno**
- the referenced line becomes the line represented by **lineno**, with the default offset.

LINE **lineno,integer**
- the referenced line becomes the line represented by **lineno**, with offset **integer**.

LINE **+integer**
- the referenced line is increased by **integer**.

LINE **-integer**
- the referenced line is decreased by **integer**.

lineno may be one of the following:

integer - representing line number **integer** of the session log.
I1 - representing the input line.
R - representing the currently referenced line.
C - representing the line containing the cursor on the 3270 screen display.
Z - representing the line immediately following the last line of the log.

If the line number specified is prior to the first existing line, the referenced line will become this first existing line; if the line specified is beyond the last existing line, the referenced line becomes one beyond the last existing line.

COL: To change the line display mode and, if the mode is "clipped", to change the referenced column or query its number.

COL [*W* | **colno** | (+|-) **integer**]

COL W
- the display mode becomes "wrapped", and the referenced column becomes 0.

COL **colno**
- the display mode becomes "clipped" and the referenced column becomes **colno**.

colno may be one of the following:

integer - representing column number **integer**
R - representing the currently referenced column
C - representing the column containing the cursor on the 3270 screen

COL + **integer**
- the display mode becomes "clipped" and the referenced column is increased by **integer**.

COL - **integer**
- the display mode becomes "clipped" and the referenced column is decreased by **integer** (to a minimum of 0).

PAGE: To increase or decrease the number of the referenced line so that the displayed page will skip forward or back an exact number of display pages.

PAGE (+|-) **integer**

PAGE + **integer**
- the referenced line is changed so that the displayed page will skip forward exactly **integer** pages, unless this would place the referenced line beyond the end of the session log, in which case the referenced line becomes the next input line.

PAGE - **integer**
- the referenced line is changed so that the displayed page will skip backwards exactly **integer** pages, unless this would place the referenced line before the first existing line, in which case the referenced line becomes this first existing line.

FORW: To search forward for a string in the session log.

FORW **string**

where **string** is a quoted string

The referenced line is increased to be the first line beyond the current one which contains the string **string**, if such a line exists. If there is such a line, the referenced line offset is set to the default. If there is no such line, the command fails, with completion code 100, and character result '* *STRING NOT FOUND*'. The referenced line and offset are not changed.

BACK: To search backward for a string in the session log.

BACK **string**

where **string** is a quoted string

The referenced line is decreased to be the first line before the current one which contains the string **string**, if such a line exists. If there is such a line, the referenced line offset is set to the default. If there is no such line, the command fails, with completion code 100, and character result '* *STRING NOT FOUND*'. The referenced line and offset are not changed.

DEL: To delete any trailing section of the session log.

DEL **lineno**
- the lines of the session log from **lineno** to the end of the log are deleted.

lineno can be:

integer	-	representing session log line number **integer**;
R	-	representing the currently referenced line;
C	-	representing the line of the session log containing the cursor of the 3270 screen.
Z	-	representing the line immediately following the last line of the log.

INS: To change the next input line by inserting a specified string, either in the next input line itself, or a copy of a session log line.

INS **charpos,string**

where **string** is a quoted string, and

where **charpos** represents a line, and a character within that line, and can be one of the following:

integer1 integer2
- representing the **integer2**'th character of session log line number **integer1**;

*I*1 integer2
- representing the **integer2**'th character of the next input line;

C - representing the line, and the character within the line, indicated by the cursor on the 3270 screen. The line may be either a session log line or the next input line. If the cursor is not on the APL part of the screen, it is taken to represent the end of the next input line. (If the cursor is not on the APL screen, *C* represents the end of the next input line).

The next input line is amended as follows:

- if **charpos** represents the next input line, the string **string** will be inserted immediately before the character indicated;
- if **charpos** represents a line of the session log, the next input line will be replaced by the result of inserting the string **string** immediately before the character indicated, in that line of the session log. The session log line itself is not altered, however.

HELP: To obtain some prompting about commands and how to use them.

HELP [**commandword**]

HELP
- the result is a list of valid session manager command words.

HELP **commandword**
- the result is an indication of the syntax of the command **commandword**.

ENTRY: To explicitly control whether or not the next input line is submitted to APL.

ENTRY F | *B*

ENTRY F - the next input line will be submitted to APL if APL is in input mode;

ENTRY B - the next input line will be prevented from being submitted to APL even if APL is in input mode.

The *ENTRY* command can be used in conjunction with the *INS* command to specify whether or not the line constructed by the *INS* command is to be submitted to APL immediately. Thus:

/INS C,')LOAD 666 *BOX'/ENTRY F*

or

/INS C,')LOAD 666 *BOX'/ENTRY B*

or in conjunction with the *LINE* command, for instance to combine a scrolling of the display with the immediate entering of the next input line. Thus:

*/LINE I*0,0*/ENTRY F*

ROLL: To set or query the default value of the referenced line offset.

ROLL [**integer**]

ROLL
- the result is the current default value of the referenced line offset.

ROLL **integer**
- the default value of the referenced line offset becomes **integer**.

PFK: To set or query the strings associated with the program function keys.

PFK [**integer**[**,string**]]

where **string** is a quoted string

PFK
- the results are the strings currently associated with all program function keys (excluding those that have empty strings associated with them). Each result is in the form:

  PFK **integer,string**

PFK **integer**
- if the string currently associated with PFKEY number **integer** is empty, there is no result; otherwise the result is that string, in the same form as above.

PFK **integer,string**
- the string associated with PFK number **integer** becomes **string**. The string can be empty.

XPFK: To issue the command or command sequence associated with a program function key.

XPFK **integer**
- the string associated with PFKEY **integer** is processed as a command or command sequence.

GET: To retrieve a block of lines from the session log.

GET **lineno1 lineno2**
- the results are lines **lineno1** to **lineno2** inclusive of the session log.

Where **lineno** can be as for the *DEL* command.

If either **lineno1** or **lineno2** are outside the limits of the session log, they will be forced to the first or last lines of the log.

(Note: If **lineno2** is *Z*, the results are the lines from **lineno1** to the end of the session log.)

The principal use of the *GET* command is to retrieve data from the session log by issuing the command via AP6. However, if issued from the 3270 screen, it has the effect of copying a block of lines to the current end of the log.

Some commands which are related exclusively to the operation of the 3270 screen are detailed later.

The 3270 Screen

The 3270 screen is divided into two areas:

- The **session manager row:** to communicate exclusively with the session manager;
- The **APL screen:** to communicate principally with APL.

The Session Manager Row:

The first row of the screen is used as follows:

- Screen indicator:

  'U' : means that the 3270 screen is "unlocked". In this state the "next turn" with the screen is the user's: the session manager will not update the screen until the user has entered a screen transaction (i.e. pressed a data or non-data entry key);

  'L' : means that the 3270 screen is "locked". In this state the "next turn" belongs to the session manager: anything entered by the user will be ignored, except the pressing of a non-data key.

- Session Log Indicator:

  'H' : ("Holding") indicates that there are lines in the session log (or all or part of the next prompt from APL) that have not been seen by the user (this normally occurs when the screen is full).

- APL Mode Indicator:

  INP : means that APL is in input mode;
  RUN : means APL is in run mode;
  KEYB : means that APL is in run mode because the APL system command *)KEYB LOCK* has been issued.

- Command area: a field for the user to enter a session manager command or command sequence, and for a result of a command to be displayed.

- "Line" Area: a field for the user to enter an argument to the *LINE* command, and for the current number of the referenced line to be displayed. The area is preceded by a protected field containing the text *LINE*.

- "Col" Area: a field for the user to enter an argument to the *COL* command, and for the current column mode (or if this is "clipped", the current number of the referenced column) to be displayed. The area is preceded by a protected field containing the text *COL*.

The screen, session log and APL mode indicators are protected fields; the command, *LINE* and *COL* areas are unprotected.

Any time that the screen is unlocked, session manager commands can be entered, either in the command area, or in the case of the *LINE* and *COL* commands, by entering just the command argument in the line or column area. Any single result, if it will fit in the command area, will be displayed there. Otherwise the command (with the characters *)SESM* prepended), and its result(s), will be appended to the session log, and hence displayed on the APL screen.

The APL screen:

The remaining rows of the screen are devoted to displaying parts of the session log and the next input line. The contents of the display are normally determined as follows:

- The lines of the session log displayed will be a block of consecutively numbered lines;

- The referenced line will be displayed starting *N*+1 rows down the APL screen (where *N* is the referenced line offset);

- The parts of each line to be displayed will be determined by the column mode ("wrapped" or "clipped"), and if the latter, by the value of the referenced column;

- If the end of the session log is reached before filling the screen, the next input line, or as much of it as will fit, will be displayed immediately following the last session log line.

Each line of the session log occupies a separate, unprotected, 3270 screen field (possibly taking up more than one row if the display is in wrapped mode); the next input line occupies a separate unprotected field, extending over several rows but not necessarily to the end of the screen. The area beyond the next input line is protected.

Preparation and Submission of Next Input Line

The next input line can be changed in several ways:

- by the use of the *INS* session manager command (note that this can even be done when the screen is locked, by using AP6);

- if the screen is unlocked, by using one or more of the following devices in a single screen transaction:

  1) altering the displayed next input line itself;

  2) altering a displayed line of the session log, leaving the cursor somewhere other than on this line;

  3) positioning the cursor somewhere on a displayed line of the session log, with or without altering the line.

The next input line will be replaced by the result of catenating the individual lines (amended or not), in the above order. In the case of 2), if more than one such line is amended, the order of catenation will give precedence to the lowest numbered line.

Note that any such preparation of the next input line can be carried out regardless of whether APL is in input or run mode. Hence, preparation in anticipation of APL's next prompt is possible. For example, unlock the screen by pressing PA2 (or its equivalent), enter the next line of input on the screen, and press ENTER to lock the screen again.

The preparation of the next input line is considered to be complete when:

- the screen is in a "resting state" (see below); and
- the screen did not "move" as a result of the last transaction (i.e. the referenced line and offset did not change); and
- a command was not entered from the session manager row; and
- a program function key was not pressed; and

- the cursor was placed somewhere on the next input line.

(In brief, everything is as stationary as possible, and the cursor is on the next input line.)

If APL is in run mode when the preparation is complete, the screen will simply be locked.

If APL is in input mode, the next input line will be submitted to APL if either:

- the command *ENTRY F* was issued; or
- the command *ENTRY B* was not issued, and preparation of the next input line is considered to be complete according to the above list of conditions.

Non-Data Entry Keys

The non-data entry keys, PA1, PA2, and CLEAR have the following meanings:

PA1 - communicates (mostly) with APL:

- when signed-on to APL:
  - if APL is in run mode: an attention is sent to APL;
  - if APL is in input mode: if the screen is not in a resting state, no action is taken; if it is, the character sequence *O*, backspace, *U*, backspace, *T* is catenated to the end of the next input line, which is immediately submitted.
- when not signed-on to APL:
  - the terminal is disconnected from the session manager (alternatively, submitting *)OFF* as the next input line when not signed-on also causes a disconnection).

PA2 - communicates with the session manager

- if the screen is unlocked: the screen is completely refreshed, and any lines at the end of the session log that have not yet been displayed, are deleted from the session log.

- if the screen is locked: the screen is completely refreshed, and unlocked. Also, any "long" session manager command that is currently being processed may be interrupted and aborted. Such commands are *GET*, *FORW* and *BACK*; i.e. those for which the amount of processing required is dependent on the amount of session log data involved.

CLEAR - the session manager command *'LINE I1,0'* is executed and the screen is refreshed. Thus, a clear APL screen, with the next input line at the top is displayed.

Note: On some terminals, when the screen is locked the only keys that can be pressed are ATTN, and SYSREQ. On such terminals, the following equivalences apply when the screen is locked:

- ATTN is equivalent to PA1
- the keystroke sequence SYSREQ, SYSREQ, ATTN is equivalent to PA2.

Updating of the APL Screen

The APL screen will be updated when lines are appended to the session log, until the screen becomes full. If there are lines not yet displayed at this point, the referenced line will be advanced to the first session log line not yet seen. Thus the next screen display will show the further lines of the session log, normally with an overlap of the few screen rows (due to the referenced line offset). This is the normal "forward moving" display of the session log, but it can be altered at any time by explicitly setting the value of the referenced line with the session manager commands *LINE*, *PAGE*, *FORW*, or *BACK*; i.e. scrolling, paging or searching can be performed at any time.

In general, therefore, the display on the APL screen may be in either:

- a resting state: the screen is displaying the last zero or more lines of the session log, and all of the next input line; or

- a paged-back state: any non-resting state.

The general rules for transition from one screen display to another are as follows:

- an explicit setting of the referenced line (and possibly the referenced line offset) will determine the contents of the next display; otherwise,

- if a command is entered on the session manager row, or a program function key is pressed, then the referenced line and offset will not change (hence, the display will not "move"), unless of course the commands entered were ones that explicitly change these quantities; otherwise,

- the referenced line and offset will be set such that the last display that was in a resting state will be re-displayed.

Thus, if nothing was entered on the session manager row, pressing ENTER will result in a display in a resting state.

The rules for where the cursor will be positioned when a screen display is generated are as follows:

- if the screen is locked: the cursor will be placed at the end of the APL mode indicator (a protected position);

- if the screen is unlocked:

- if, as part of the preceding screen transaction, the user entered a command argument into the *LINE* or *COL* areas of the session manager row, the cursor will be placed in that area; otherwise,

- an attempt will be made to position the cursor in the next input line: if the next input line is the result of an *INS* command, at the end of the insertion; otherwise at the end of the next input line; if this position is not on the screen:

- the cursor will be positioned on the session manager row, at the start of whichever of the command, *LINE* or *COL* areas was last used as part of a screen transaction.

In other words, the cursor will tend to stay in the *LINE* or *COL* areas as long as one of them is used. Failing that, it will try to find a sensible place in the next input line; failing that, it will resort to whichever part of the session manager row was most recently popular.

The *ROLL* Setting

The *ROLL* setting, a non-negative integer less than the number of rows on the APL screen, specifies the default value of the referenced line offset. Hence, it controls the amount of screen overlap in normal "forward moving" display of the session log. It can be set by means of the *ROLL* session manager command.

The *AUTOPAGE* Setting

When the screen becomes full during the normal forward moving display of the session log, the user can control whether or not the screen is to be immediately updated with the next display. The *AUTOPAGE* setting controls this decision:

- *OFF*: when the screen becomes full, an *H* will be displayed in the session log indicator, and the screen will unlock, giving the user control over when to display the next page. (Note: this will not happen if the last session log line on the APL screen is input to APL);

- *ON*: when the screen becomes full, it will not unlock, but will immediately display the next page.

The *AUTOPAGE* setting can be controlled by means of the *AUTOPG* session manager command.

The Overstrike Escape Character

It is possible to enter certain characters, for which there is no key on the terminal, as a sequence of 3 characters: an "overstrike escape character" (OEC), and 2 other characters defined at a particular installation to be the "overstrike" combination which replaces the unavailable character. For instance, if the OEC is ¯, the sequence ◇ might be defined to be the APL "diamond" character.

The OEC can be defined by means of the *OEC* session manager command.

Input Line Overlaying

When the display is in a paged back state, the next input line cannot be displayed. In this state, it is possible to specify that the bottom n rows of the APL screen be devoted to display of the next input line. This affords the user the convenience of being able to amend the line whilst in this state. This device is available through the *INP* session manager command.

Colour/Highlight Attributes

The colour and highlight attributes of the various fields on the screen can be controlled by means of the *FIELD* session manager command. There are a number of distinguished **field types**:

MODE	- the screen, session log, and APL mode indicators
TITLE	- the (protected) *LINE* and *COL* titles
PARMS	- the *LINE* and *COL* areas, and the command area.
REFLINE	- the referenced line
UNSEEN	- when scrolling forward, the first line of the session log which was not displayed on the previous display; when scrolling backward, the last line that was not displayed on the previous display
SESM	- a line of the session log that was input to or output from the session manager
INOUT	- a line of the session log which was input to APL
OUT	- a line of the session log which was output from APL
PROMPT	- a "false" APL prompt (i.e. that part of the prompt which was produced by using *⎕ARBOUT* '')
INPUT	- the next input line

In the case of a 3270 terminal with extended features, the colour and highlight attributes of these field types can be specified (including a specification of "neutral"); for a terminal without extended features, each field type may be specified to be either "high" or "neutral". If either of the attributes for field type *REFLINE* are specified to be "neutral", that attribute will default to that of *SESM*, *INOUT* or *OUT* as appropriate. The same is true for field type *UNSEEN*. The specifications for *REFLINE* take precedence over those for *UNSEEN* if a line falls in both categories.

3270 Session Manager Commands

The following session manager commands are related specifically to the 3270 screen:

AUTOPG: To control or query the autopage setting.

```
AUTOPG [ON|OFF]
```

AUTOPG
- the result is the current value of the autopage setting ("on" or "off").

AUTOPG ON - the autopage settings becomes "on".

AUTOPG OFF - the autopage setting becomes "off".

FIELD: To specify or query the colour/highlight attributes of the distinguished screen field types.

FIELD [**fldtype**[,[**colour** [**highlight**]]]]

where **fldtype** is one of the distringuished field types;
colour is *NEUTRAL* or one of the colours:

BLUE, RED, PINK, GREEN, TURQUOISE, YELLOW, WHITE

or, in the case of a terminal without extended features:

HIGH;

highlight is *NEUTRAL* or one of:

BLINK, REVERSE, UNDERLINE

FIELD
- the results are the colour/highlight settings for all field types, each one in the form:

FIELD **fldtype,colour highlight**

FIELD **fldtype**
- the result is the colour/highlight setting for the field of type **fldtype**, in the above form.

FIELD **fldtype,**
- the colour/highlight setting for the field of type **fldtype** becomes *NEUTRAL/NEUTRAL*.

FIELD **fldtype,colour**
- the colour setting for the field of type **fldtype** becomes **colour**.

FIELD **fldtype,colour highlight**
- the colour/highlight setting for the field of type **fldtype** becomes **colour/highlight.**

INP: To take advantage of the "input overlaying" device.

INP **integer**

- if the screen is in a paged back state, the bottom **integer** rows of the screen will be devoted to display of the next input line. Amendments to this line can then be made in the usual way. The effect of this command persists through changes in the referenced line as long as the screen remains in a paged back state, but is cancelled as soon as the screen returns to a resting state.

OEC: To establish (or revoke) the overstrike escape character.

OEC [*OFF*|**character**]

where **character** is a single quoted character,

OEC
- the result is the current OEC, or *OFF* if there is none.

OEC OFF
- the current OEC is revoked, so that no OEC is defined.

OEC **character** - the OEC becomes **character**.

The Session Manager Command Auxiliary Processor (AP6)

The session manager auxiliary processor allows the APL task to which the session is attached to issue session manager commands from an APL program. The most immediate use of this AP is to set the various session parameters (referenced line offset, *AUTOPG* setting), the program function keys, and the field colour/highlight attributes with an APL function, but there are other possible applications such as automated browsing through the session log.

The processor id of the AP is 6, and communication with the APL task is via 1 or 2 shared variables. The name of the first variable must begin with the characters *CTL*, and the second with *DAT*. If 2 variables are used, their names must be identical after the first 3 characters. (The only reason to use 2 variables is if an application has problems with space in the workspace.)

Communication with the AP via a *CTL* variable only is achieved by sending a chain of command sequences as follows:

CTL ← (0 ⊃ *CMDSEQ*1),(0 ⊃ *CMDSEQ*2), ... ,(0 ⊃ *CMDSEQ*(*N*))

where each *CMDSEQ* is a character vector consisting of a session manager command sequence. The sequences in the chain will be executed by the session manager one after the other, up to the first element in the chain which is not of the form 0 ⊃ *CMDSEQ*. The command sequence in a chain element will be executed by the session manager up to the first command in the sequence which gives a non-zero completion code.

On completion of processing, *CTL* will be set to:

CRC ⊃ (<*ERC*1 ⊃ *ERES*1),(<*ERC*2 ⊃ *ERES*2), ... ,<*ERC*(*M*) ⊃ *ERES*(*M*)

where:

- *M* is such that the (*M*+1)th chain element was the first invalid element

- *CRC* is 0 if all chain elements were valid; otherwise, a positive integer code indicating the problem with the first invalid chain element. The meanings of these codes are given in the section titled "Miscellaneous Messages";

- *ERC*(*I*) is the completion code of the last command executed in the *I*'th command sequence.

- *ERES*(*I*) consists of the results of the commands in the *I*'th sequence:

  (<*CRES*1),(<*CRES*2), ... ,<*CRES*(*S*)

  where *CRES*(*J*) consists of the results of the *J*'th command in the *I*'th command sequence:

  *RES*1 ⊃ *RES*2 ⊃ ... ⊃ *RES*(*C*)

  Each such *RES*1 is a character vector in which the first 2 characters are control information as follows:

  The 1st character is an encoding of the type of the result, ⎕*AV*[⎕*IO*+2⊥*LOGF*,*SSMF*,*INPF*], where:

  LOGF - 0 means the result is a line from the session log; 1 means it is not;

  and if *LOGF* ↔ 0:

  SSMF - 0 means the line was to/from APL; 1 means the line was to/from the session manager;

  INPF - 1 means the line was input to APL or the session manager; 0 means it was output.

  If the result is a session log line which was input to APL in which the prompt resulted from a use of ⎕*ARBOUT* '', the 2nd character is ⎕*AV*[⎕*IO*+*L*], where *L* is the length of this prompt.

Communications with the AP via *CTL* and *DAT* variables is achieved as follows:

DAT ← *CMDSEQ* ⋄ *CTL* ← 0

where *CMDSEQ* is a (character vector) command sequence as above.

On completion of processing, *DAT* will be set to *ERES* as above, and *CTL* will be set to *ERC* as above.

Note: Workspace 5 *IBM3270* contains a function *IDSHPROF*, which is an example of the use of AP6 to establish PF key definitions, etc.

Miscellaneous Messages

The non-zero completion codes for commands submitted to AP6 are:

```
  1  - invalid command word
  2  - invalid parameter to a command
  3  - command recursion too deep
  4  - command aborted by user
  5  - result too big

100  - string not found: FORW, BACK
101  - next input line too long: INS
102  - PFK definition too long: PFK
```

The codes resulting from incorrect chain elements submitted to AP6 are:

```
 11  - CTL rank error
 12  - CTL length error
 13  - CTL domain error
 21  - DAT rank error
 22  - DAT length error
 23  - DAT domain error
 41  - no DAT variable value
 42  - output buffer overflow
```

The following messages from the session manager may be appended to the log:

APL NOT RESPONDING
This usually means either that APL is not running, or that resources required for the session manager to connect to APL are not available

NO SVP
A fundamental required resource, the SHARP APL shared variable processor, is not available.

Note: the spelling of these messages can be altered by a site.

In addition, it is possible for the session manager to issue PA messages to all its users.

There are three full screen management interfaces.

AP124 is the simplest and most efficient and should be adequate for most applications. It uses a CTL/DAT type shared variable interface.

AP126 is an interface to the IBM product GDDM (Graphical Data Display Manager) and provides all of the facilities of AP124 plus extensive colour graphics capabilities. It uses a CTL/DAT type shared variable interface.

The ARBIN Manager is the lowest level interface and gives the user **complete** control of the device. It uses the *□ARBIN* and *□ARBOUT* primitives to send and receive arbitrary data streams. In addition the ARBIN manager supports program control of some parts of the Session Manager.

Using AP124, an application may define and manipulate rectangular areas called SCREEN FIELDS, or FIELDS. In addition to area, a field has attributes, such as highlighting, input protection, or color. Area and attributes can be set during the initial field definition and reset afterward using full screen commands. These specifications are made in the form of a format matrix (discussed later).

Once the appropriate variables have been shared with AP124, an application can assign operation codes directly, or through the use of cover functions contained in one of the full screen management workspaces in public library 5. Full screen commands can be grouped into three areas, namely

1) Operations to define fields
2) Operations to write and read
3) Miscellaneous operations

AP124 allows an application to pass full screen operations and data using a pair of shared variables (prefix CTL and DAT) or in a single CTL variable if the data is in enclosed form. Below is a list of operation codes and their descriptions.

CTL	DAT	Description
0	*	no operation
1	fmt matrix	format screen into fields
1,fld	fmt matrix	reformat screen into fields
2,fld	data	write to screen immediately
3	*	read from screen and wait
4,fld	data	buffered write to screen
5,fld	*	get data
6,fld	type	change field type
7,fld	intensity	change field intensity
9	*	read format matrix
11	*	sound alarm
12	position	set cursor position
16	attribute	change field attributes
20	*	erase the screen
24,fld	color	change field color
26,fld	highlight	change field highlighting
28	*	return raw data stream for this screen
30	*	get data for modified fields
31	*	get data for modified fields but in enclosed form
35,fld	character attributes	buffered write of character attributes
37,fld	character attributes	immediate write of character attributes
40	*	get screen data
41	*	get screen attributes

Several pairs of variables may be shared simultaneously with AP124 allowing several independent logical screens. To allow this, full screen variable naming conventions have been adopted. The first three letters of the control variable must be CTL, and similarly, DAT for the data variable. Each variable may be up to eleven characters long and are paired by suffix to identify a logical screen. Since AP124 has no outstanding variable offers, the degree of coupling will be:

1 - after the application's initial offer
2 - when AP124 accepts the offer

For operations requiring both a control variable and a data variable, the data variable is assigned first. A non zero return code in the control variable means that an error occurred during the operation. Below is a list of return codes and their descriptions.

```
  0 NORMAL RETURN -  OPERATION SUCCESSFUL
 11 CONTROL VARIABLE RANK ERROR
 12 CONTROL VARIABLE LENGTH ERROR
 13 CONTROL VARIABLE DOMAIN ERROR
 14 INVALID COMMAND
 15 REQUEST TO POSITION CURSOR IN UNDEFINED FIELD
 21 DATA VARIABLE RANK ERROR
 22 DATA VARIABLE LENGTH ERROR
 23 DATA VARIABLE DOMAIN ERROR
 24 DATA VARIABLE NOT SHARED
 30 INVALID FIELD NUMBER
 32 DEFINED FIELD EXTENDS BEYOND THE SCREEN
 33 REFERENCE OUTSIDE FIELD DEFINITION
 35 LIGHT PEN FIELD STARTS IN COLUMN 1
 36 LIGHT PEN FIELD (HEIGHT 1) NOT CONTAINED
    IN 1 PHYSICAL SCREEN LINE
 37 INVALID FIELD TYPE
 38 INVALID FIELD INTENSITY
 99 UNKNOWN 3270 DEVICE ERROR
201 INVALID COLOR
202 INVALID HIGHLIGHT
204 INVALID PROGRAM SYMBOL SET
205 INTERNAL BUFFER OVERFLOW
206 INVALID SKIP COLUMN IN FORMAT MATRIX
207 INVALID NULLS COLUMN IN FORMAT MATRIX
```

An application may request many full screen operations with a single shared variable use by passing several CTL and DAT values together as an enclosed vector. In this case, the CTL control variable is used for both input and output, and the DAT data variable need not be shared. When the CTL argument is enclosed, AP124 automatically returns an enclosed result, consisting of an enclosed return code element, followed by enclosed data elements resulting from full screen operations and normally found in DAT. When disclosed, the return code element is simple and has two numbers, specifying completion code (as

discussed earlier), and the number of operation codes processed. Processing terminates when an error occurs, so in the event of a non zero completion code, the second number indicates how many operation codes were sucessfully processed.

1) OPERATIONS TO DEFINE FIELDS

FORMAT the SCREEN

CTL=1
DAT=format matrix

The format matrix is numeric with a row describing each screen field. Columns specify starting row, starting column, depth, width, type, intensity, color, program symbol set, highlighting, skip, and trailing blank processing. A format matrix with fewer than eleven columns assumes default values from the following default format matrix.

1,1,device rows,device cols,2,1,0,0,0,0,1

a) TYPE is:
- 0 - character input/output allowed
- 1 - numeric character input/output allowed
- 2 - character output only (default)
- 3 - character output/light pen interruptable
- 4 - character output/light pen selectable

A field type of 2, 3, or 4 causes that field to be "protected", so data cannot be entered by the user.

b) LIGHT PEN FIELDS

Terminals with light pen support can have fields defined as being light pen sensitive as well as "normal" fields. A screen is "mixed" if it contains both light pen and non-light pen fields. The two types of light pen fields are "selectable" and "interruptable", and a light pen interruptable field can return a "light pen" or "enter" completion code.

All light pen fields must have a designator character written in the first column of the data. Also, the attribute and designator characters must appear on the same screen row (so light pen fields may not begin in column one). A maximum of twelve light pen fields, or fourteen mixed fields may precede the final light pen field on any row.

SELECTABLE FIELDS

In the case of selectable fields, one of two designator characters may be used: question mark (?) or greater-than (>). The ? indicates that the selectable fields has not been modified, and the greater than sign indicates that a field has been modified. Therefore an application may pre-select a field by initially writing a column of >'s. A selectable field switches between unmodified and modified each time the light pen is applied. If the designator character is illegal or omitted, AP124 changes the first column to "?" (unmodified). A field whose designator character is ">" upon input completion, is considered to have been modified.

INTERRUPTABLE FIELDS

An interruptable field has a designator character of blank or ampersand (&) (types 1 and 2 respectively). When a light pen is applied to a type 1 field, input is completed, a light pen completion code is returned, and field numbers for modified selectable fields (modified non-light pen fields) are ignored. Type 2 interruptable (3276, 3278, 3279 only) fields return a completion code identical to that of "enter", and all modified field numbers, so it behaves as if the user had pressed the "enter" key. An illegal or omitted designator character is changed to a blank character.

c) INTENSITY is:
- 0 - do not display data
- 1 - normal intensity (default)
- 2 - high intensity

d) COLOR is:
- 0 - green (default)
- 1 - blue
- 2 - red
- 3 - pink
- 4 - green
- 5 - turquoise
- 6 - yellow
- 7 - white

e) PROGRAM SYMBOL SET: must be zero (default)

f) HIGHLIGHTING is one of:
- 0 - no highlighting (default)
- 1 - blink
- 2 - reverse video
- 4 - underscore

g) SKIP is:
- 0 - skip to next unprotected field when the cursor is advanced past the end of the current unprotected field (default)
- 1 - allow the cursor to move into the next screen area regardless of whether it is protected or not protected

h) BLANK PROCESSING is one of:
- 0 - nulls read as blanks, trailing blanks write as blanks
- 1 - nulls read as blanks, trailing blanks write as nulls
- 2 - nulls read as nulls, trailing blanks write as blanks
- 3 - nulls read as nulls, trailing blanks write as nulls

Before and after each row of each field, is an attribute character defining the following screen area. These attribute characters occupy a physical space, so allow for this when designing a panel (don't include this space as part of the width specification). When the format matrix SKIP column is zero (default), undefined areas are skipped when the cursor moves there during input.

REFORMAT the SCREEN

CTL=1,field numbers
DAT=format matrix

Used to reformat selected fields, the format matrix is identical to that defined in the FORMAT section except that each row now has a corresponding field number. Existing data is not erased during reformatting.

READ the FORMAT MATRIX

CTL=9
DAT=(none)

The result is the format matrix used to define the current screen.

SET FIELD TYPE

CTL=6,field numbers
DAT=type numbers

Field types for selected fields are reset.

SET FIELD INTENSITY

CTL=7,field numbers
DAT=intensity numbers

Intensities for selected fields are reset.

SET FIELD COLOR

CTL=24,field numbers
DAT=color numbers

Colors for selected fields are reset.

SET FIELD HIGHLIGHTING

CTL=26,field numbers
DAT=highlighting numbers

Selected field numbers' highlighting is reset.

SET FIELD ATTRIBUTES

CTL=16,field numbers
DAT=attribute numbers

Attribute numbers range from zero to seven and control the SKIP and BLANK PROCESSING bits described in the FORMAT section. Attribute values are:

0 - nulls read as blanks,
trailing blanks write as blanks, no autoskip

1 - nulls read as blanks,
trailing blanks write as blanks, autoskip

2 - nulls read as blanks,
trailing blanks write as nulls, no autoskip

3 - nulls read as blanks,
trailing blanks write as nulls, autoskip

4 - nulls read as nulls,
trailing blanks write as blanks, no autoskip

5 - nulls read as nulls,
trailing blanks write as blanks, autoskip

6 - nulls read as nulls,
trailing blanks write as nulls, no autoskip

7 - nulls read as nulls,
trailing blanks write as nulls, autoskip

2) OPERATIONS TO WRITE AND READ

BUFFERED WRITE

CTL=4,field numbers
DAT=data

Data written using this operation goes into the screen's buffer and is displayed during the next physical write. The data is a character matrix and each row is written to the row whose number is in the corresponding position in the list of field numbers. DAT may be a character vector if only one field is being written.

BUFFERED WRITE ATTRIBUTES

CTL=35,field numbers
DAT=attributes

Attributes of characters in a field are specified. The relationship between 'field numbers' and the DAT argument is the same as for 'BUFFERED WRITE', but instead of specifying the characters to display, DAT specifies the attributes of the characters.

The character attributes are encoded in the 8 bits in a character as:

SSSCCCHH

where S is for program symbol set (unsupported), C is for colour, and H is for highlight.

Colour values are:
0- default
1 - blue
2 - red
3 - pink
4 - green
5 - turquoise
6 - yellow
7 - white

Highlight values are:
0 - default
1 - blinking
2 - reverse video
3 - underscore

The attribute character is $\square AV[\square IO+(COLOR\times 4)+HIGHLIGHT]$.

IMMEDIATE WRITE

CTL=2,field numbers
DAT=data

DAT is a character matrix as defined for the buffered write operation, except that it is displayed immediately.

IMMEDIATE WRITE ATTRIBUTES

CTL=37,field numbers
dat=attributes

As for 'BUFFERED WRITE ATTRIBUTES', except that it causes the screen to be written immediately.

READ and WAIT

CTL=3
DAT=(none)

This operation causes a physical write to the screen of any buffered data, then waits for the user to complete input. The resulting DAT contains a completion code indicating how input was caused, and, depending on the completion code, a modified and/or list of modified field numbers as defined below.

USER ACTION	*COMPLETION CODE*	*MODIFIER*	*CURSOR POSITION FIELD*	*ROW*	*COLUMN*	*FIELDS*
ENTER KEY	0	0	*FLDNUM*	*ROW*	*COL*	*FLDNUMS*
PF KEYS	1	1-24	*FLDNUM*	*ROW*	*COL*	*FLDNUMS*
LIGHT PEN	2	0	*FLDNUM*	*ROW*	*COL*	*FLDNUMS*
PA KEYS	4	1-3	-	-	-	-
CLEAR KEY	5	-	-	-	-	-
NO INPUT	6	-	-	-	-	-

From the result, an application can ascertain if and how input was caused and what fields were modified.

GET FIELD DAT

CTL=5,field numbers
DAT=(none)

The result contains a character matrix where a row contains data for the field number in the corresponding position of the field number list.

GET MODIFIED DAT

CTL=30
DAT=(none)

Data is returned in DAT for fields which were modified during the last READ AND WAIT operation.

GET MODIFIED DAT EXTENDED

CTL=31
DAT=(none)

As for GET MODIFIED DAT except that the data is returned as a vector of enclosed elements. Each element's data has been stripped of trailing blanks.

GET SCREEN

CTL=40
DAT=(none)

The result is a character matrix the same size as the screen (rows by columns) containing the data for this screen.

GET SCREEN ATTRIBUTES

CTL=41
DAT=(none)

The result is a character matrix the same size as the screen (rows by columns) containing the character attributes, encoded in the same as for the 'WRITE ATTRIBUTE' operation, for this screen.

3) MISCELLANEOUS OPERATIONS

ERASE THE SCREEN

CTL=20
DAT=(none)

The full screen display is cleared.

SET CURSOR POSITION

CTL=12
DAT=position

Position is field number, row, and column. After the next write operation, the cursor is placed at the row and column of the selected field number. When the field number is zero, the row and column are taken from the screen's upper left hand corner. The cursor position is automatically updated when a field is modified by the user.

SOUND THE ALARM

CTL=11
DAT=(none)

The audible alarm is sounded after the next physical write.

At present, there are three public workspaces located in public library 5 to aid a full screen application in the use of AP124. Each one uses the same basic operation codes, but differ in other ways. The three workspaces are:

```
5 IBM3270
5 AP124
5 AP124E
```

Following is a description of each workspace.

Workspace 5 *IBM*3270

This workspace contains functions with the same names as those supplied with IBM's AP124x, so that an applications programmer, used to working with IBM's functions will feel comfortable using this workspace. In addition, this workspace contains functions to control color, and highlighting, as well as extended capabilities peculiar to SHARP APL's AP124.

FUNCTION NAME AND SYNTAX	CTL	DESCRIPTION
FORMAT fmtmatrix	1	fmtmatrix is the format matrix containing four, six, or eleven columns.
fld *REFORMAT* fmtmatrix	1	Format matrix fmtmatrix, is applied to field numbers fld.
R←READFORMAT	9	The result is the current format matrix.
fld *FIELDTYPE* typ	6	Field numbers in fld are reset to the values in typ.
fld *INTENSITY* int	7	Field numbers in fld are reset to the values in int.
fld *PSS* ps	-	Field numbers in fld will use the program symbol sets in ps (currently unsupported).
fld *COLOR* col	24	Color values of field numbers in fld are reset to col.
fld *HIGHLIGHT* hi	26	Highlight values for fields in fld are reset to hi.
fld *FIELDATTR* att	16	Attribute values in att are applied to field numbers in fld.

fld *WRITE* data	4	Data in a row of character matrix data is written to the buffer of the field whose number is in the corresponding position of fld.
fld *WRITEATTR* att	35	Encoded character attributes in a row of att are written to the buffer of the field whose number is in the corresponding position of fld.
fld *IMMWR* data	2	As for *WRITE* except that the data is written to the screen immediately.
fld *IMMWRATTR* att	37	As for *WRITEATTR* except that the attributes are written to the screen immediately.
R←READSCREEN	3	Buffered data is written to the screen and the full screen manager waits for the user to complete input.
R←GETFIELDS fld	5	Data as a character matrix is returned for field numbers fld.
R←READDATA	30	Data in character matrix form is returned for fields which were modified during the last *READSCREEN* operation.
R←READDATAX	31	As for *READDATA* except that the data is returned as a vector of enclosed elements.
R←GETSCREEN	40	A character matrix representing the current characters on the screen is returned.
R←GETSCREENX	41	A character matrix representing the current character attributes on the screen is returned.
ERASESCREEN	20	The full screen display is erased.
fld *SETCURSOR* rowcol	12	The cursor is placed at the row and column of rowcol in the field number fld.
ALERT	11	The alarm is sounded.

STATE	ARB	*STATE* reports the session manager status including the number of □*ARBIN* screens, first and last session manager line numbers, and screen dimensions. This function uses the □*ARBIN* interface.

Workspace 5 *AP*124

This workspace contains all of the shared variable functions of workspace 5 *IBM*3270 and none of the □*ARBIN* functions. The functions are in stripped down form and assume that variables *CTLS* and *DATS* are shared and fully coupled, otherwise an error message results. The function *SHAREAP*124 shares these variables and returns a 1 when coupling is complete.

Workspace 5 *AP*124*E*

Functions in this workspace are identical to those of workspace 5 *AP*124 except that they use the COMMAND CHAINING feature. Full screen operations are stored in a workspace resident buffer until being forced out by the application. The letter E (for Enclosed array) prefixes all function names so that they may be distinguished from those of the other workspace. Although all commands are buffered and are in enclosed array form, the buffer contents may optionally be sent after each operation. The functions *EFRCON* (turn on automatic buffer force) and *EFRCOFF* (turn off automatic buffer force) set the appropriate state which may be checked using the function *ESTATE*. The workspace buffer is cleared after its commands are sent, but it may be cleared by the application at any time using the function *ECLRBUF*. See *DESCRIBE* in this workspace for a more detailed description of its operation.

AP126 provides a CTL/DAT type shared variable interface to the IBM product GDDM (Graphical Data Display Manager). AP126 provides all of the facilities of AP124 plus extensive colour graphics capabilities. If an application doesn't require colour graphics, then it is better to use the simpler and more efficient AP124.

The key concepts in AP126 (e.g.,field and format matrix) are essentially the same as in AP124, and familiarity with AP124 will help in understanding AP126.

AP126 calls functions provided by GDDM. These functions are documented in the IBM manuals **Graphical Data Display Manager User's Guide** and **Presentation Graphics Feature User's Guide.**

AP126 can use any GDDM or PGF call except SPINIT, FSINIT, FSRNIT, FSTERM, FSEXIT, and CHART.

To request a GDDM call, you must specify the CTL variable with the AP126 request code that corresponds to the GDDM call mnemonic. The following table contains the AP126 request codes for each of the GDDM and PGF functions.

GDDM Control Requests

ASREAD	101	FSSHOW	105	FSALRM	109
FSFRCE	102	FSQERR	107	FSQDEV	110
FSREST	103	FSTRCE	108	ASTYPE	111
FSSAVE	104				

GDDM Symbol Set Requests

GSDSS	201	PSRSV	206	PSQSS	211
GSLSS	202	GSRSS	207	SSQF	212
PSDSS	203	PSRSS	208	SSREAD	213
PSLSS	204	GSQNSS	209	SSWRT	214
PSLSSC	205	GSQSS	210		

GDDM Page Requests

FSPCLR	301	FSPQRY	304	FSQUPG	307
FSPCRT	302	FSPSEL	305		
FSPDEL	303	FSQCPG	306		

GDDM Alphanumeric Requests

ASDFLD	401	ASFMOD	412	ASCGET	422
ASDFMT	402	ASFOUT	413	ASCHLT	423
ASDTRN	403	ASFPSS	414	ASCPUT	424
ASFCLR	404	ASFTRN	415	ASCSS	425
ASRFMT	405	ASFTYP	416	ASMODE	426
ASDFLT	406	ASRATT	417	ASQCOL	427
ASFCOL	407	ASQFLD	418	ASQHLT	428
ASFEND	408	ASQMAX	419	ASQSS	429
ASFHLT	409	ASQMOD	420	ASFCUR	430
ASFIN	410	ASCCOL	421	ASQCUR	431
ASFINT	411				

GDDM Graphics Requests

GSCLP	501	GSMIX	518	GSQCEL	535
GSFLD	502	GSMS	519	GSQCLP	536
GSPS	503	GSPAT	520	GSQCM	537
GSVIEW	504	GSARC	521	GSQCOL	538
GSWIN	505	GSAREA	522	GSQCP	539
GSCLR	506	GSCHAP	523	GSQCS	540
GSSCLS	507	GSCHAR	524	GSQCUR	541
GSSDEL	508	GSENDA	525	GSQLT	542
GSSEG	509	GSLINE	526	GSQLW	543
GSCA	510	GSMARK	527	GSQMAX	544
GSCB	511	GSMOVE	528	GSQMIX	545
GSCD	512	GSMRKS	529	GSQMS	546
GSCM	513	GSPLNE	530	GSQPAT	547
GSCOL	514	GSVEC	531	GSQPS	548
GSCS	515	GSQCA	532	GSQVIE	549
GSLT	516	GSQCB	533	GSQWIN	550
GSLW	517	GSQCD	534		

GDDM Print Requests

FSCLS	601	FSLOG	603	GSCOPY	605
FSCOPY	602	FSOPEN	604		

PGF Control Functions

CHSTRT	703	CHRNIT	704	CHTERM	705

PGF Chart Format Definition

CHAREA 721	CHYTTL 738	CHGATT 754
CHBATT 722	CHXRNG 739	CHXDTM 755
CHCGRD 723	CHYRNG 740	CHYDTM 756
CHHMAR 724	CHXINT 741	CHDATT 757
CHVMAR 725	CHYINT 742	CHNOTE 758
CHKEY 726	CHXTIC 743	CHNOFF 759
CHKATT 727	CHYTIC 744	CHNATT 760
CHKEYP 728	CHXSCL 745	CHCOL 761
CHHATT 729	CHYSCL 746	CHLT 762
CHHEAD 730	CHLATT 747	CHMARK 763
CHXSEL 731	CHXLAB 748	CHPAT 764
CHYSEL 732	CHYLAB 749	CHGAP 765
CHXSET 733	CHXMTH 750	CHGGAP 766
CHYSET 734	CHYMTH 751	CHVCHR 767
CHAATT 735	CHXDAY 752	CHPIER 768
CHTATT 736	CHYDAY 753	CHVATT 769
CHXTTL 737		

PGF Chart Drawing

CHDRAX 790	CHHIST 794	CHPIE 796
CHPLOT 791	CHNUM 794	CHVENN 797
CHSURF 792	CHBAR 795	CHSET 798

The AP126 request code and numeric parameters are set in CTL and any character data is set in DAT. A return code and numeric results are returned in CTL and any character results are returned in DAT.

More than one GDDM call can be specified in a single specification of DAT and CTL. The data for the requests are simply catenated together and set into DAT and CTL as appropriate. The results come back catenated in the same form.

The result in CTL is at least four elements for each request. The four elements are:

1. Return code: 0 - success
   1 - AP126 error
   4,8 or 12 - GDDM error
2. Reason code, or 0 if the return code was 0.
3. The number of numeric results returned in CTL for this request.
4. The length of character data returned in DAT for this request.

AP126 Service Requests

In addition to GDDM calls, AP126 will also process "service requests". Service requests are indicated by negative numbers.

Service Request	Code
Query GDDM calls	¯1
Set Error Threshold	¯2
Set EBCDIC Translation	¯4
Set AP Options	¯6
Query AP Options	¯7

Query GDDM Calls

CTL Input	¯1
CTL Output	rc vector, AP 126 request codes
DAT Output	GDDM request code mnemonics

Returns the available GDDM calls and the corresponding AP 126 request codes. The request codes are catenated onto the 4-element result and the 8 character GDDM call names are returned in *DAT*.

Set Error Threshold

CTL Input	¯2,threshold value
DAT Output	rc vector

This request sets a threshold value for GDDM error severities. When a GDDM error is encountered that has a severity greater than or equal to the threshold value, AP126 terminates processing the current string of requests. This threshold applies only to GDDM requests.

A valid threshold value is any non-negative integer. The default threshold value is 8, which causes processing to continue when a warning is encountered.

Set EBCDIC Translation

CTL Input	¯4,trans value
DAT Output	rc vector

Normally AP126 will translate all characters in a request from APL internal representation to EBCDIC and from EBCDIC to APL internal representation in a result. However, the character data parameter of SSREAD, SSWRT, PSDSS, GSDSS, and the table value parameters and character results for ASDTRN will not be translated by AP126. That is, the symbol set values, and translate table values are not translated.

This request enables you to set the translate value off, resulting in no translation by AP126, or to set the value on again. Trans value can be one of the following:

0 = set translation off
1 = set translation on (the default)

Set AP Options

CTL Input	¯6,count,AP options vector
DAT Output	rc vector

This request sets the three control requests described above in one request. A 1- to 3-element vector of options must be specified. A negative one (¯1) entry in any position in the vector retains the current value being used (it may be the default). Elements of the options vector are as follows:

1. threshold value
2. key
3. trans value

Query AP Options

Numeric Input	¯7,count
Numeric Output	rc vector,AP options vector

This request enables you to query the AP options, as described above. A numeric vector is returned containing **count** elements in the following order:

1. threshold
2. key
3. trans value

AP126 CTL Return and Reason Codes

Return	Reason	Description
0	0	No error.
1	7	Data variable length error.
1	11	Control variable rank error (must be a vector).
1	12	Syntax error in request.
1	13	Control variable domain error (must be numeric).
1	14	Invalid request code.
1	21	Data variable rank error (must be a vector).
1	23	Data variable domain error (must be characters).
1	24	Data variable not shared.
1	34	User not authorized to use this request.
1	41	Data variable has not been specified.
1	53	CTL or DAT variable too large.
1	60	GDDM not available.
1	61	Invalid parameter for AP control request.
1	62	Invalid count, code, or length on GDDM request.
1	63	Hardcopy translate not available.
1	64	Reserved page identification.
1	65	Invalid hardcopy destination.
1	66	Hardcopy destination unavailable.
1	67	Hardcopy destination already open.
4, 8 or 12	nnn	Return code is the GDDM severity code and nnn is the GDDM error code.

This section describes how an application can make complete use of the IBM 3270 device's capabilities through the primitives *⎕ARBIN* and *⎕ARBOUT*. The standard screen uses only a subset of the device's capabilities. A user can construct a raw data stream as outlined in the IBM manual GA27-2749 (IBM 3270 Component Description) and transmit it using either of the primitives. In this way an application program has control over screen format, highlighting, color, intensity and graphics (if the device is so equipped). The IBM data stream, exactly as described in the IBM manuals, can be used, but in SHARP APL the data stream has been simplified by introducing SHARP Order (SO) codes. In addition to attribute control, SOs allow an application program to:

- Define a logical screen
- Define logical screen fields on any logical screen
- Write to any logical screen or logical screen field
- Read from any logical screen or field

Constructing a Data Stream

This section outlines the formation of a valid IBM 3270 data stream with emphasis on the SHARP Order extensions.

1) The Header

A data stream consists of a character vector containing IBM data stream commands and data (for write and read commands only). To the front of this data stream is added 24 bytes of control information which tell the device-controlling software what area of terminal control should be addressed. This header currently includes a command and a logical screen number (the remaining bytes are reserved). When formed, the header and data stream are transmitted as a character or integer vector using *⎕ARBIN* or *⎕ARBOUT*. A typical transmission might look like:

```
R←⎕ARBIN HDR,DS
```

where *HDR* is the 24 byte character vector and *DS* is the data stream. The header determines how the data stream is to be applied. The first byte of the header, the command, must be one of:

Command	□AV	Meaning
FSMINQ	0	Device inquiry.

Position	Meaning
0	IDSH version
1	reserved
2	graphic escape character
3	number of rows
4	number of columns
5	unused
6	number of logical screen pages
7	extended capabilities (0 or 1)
8	PFK 13-24 mapped to 1-12
9-11	reserved
12-15	first session log line number (256 base)
16-19	last session log line number (256 base)
20-27	VTAM device ID (Z-code)

The result may be extended in the future.

These values are decoded by the function *STATE* in workspace 5 *IBM*3270.

FSMWRT	1	The data stream contains information to be written to the screen.
FSMRD	2	The data stream contains information to be written to the screen. After the write, wait for the user to cause input (ENTER key, PA1, PA2, CLEAR or any PFK) and return the input to the application program. This information includes termination type (ENTER, PFK, etc.), cursor position (row and column) and data. Data is returned in IBM EBCDIC device codes.
FSMSWCH	5	Switch logical screens. *FSMSWCH* is followed by □*AV*[**logical screen number**]. The logical screen image is copied from the current screen to that logical screen number.
FSMRDT	6	Read and Translate (same as *FSMRD* except the data is translated). Data that is read from the screen is translated into Z-codes using the default device translate table. The advantage of *FSMRDT* over *FSMRD* is that the application does not have to do the translation from device codes to Z-codes after a read.
FSMWRS	7	Write to Screen file. The full screen image currently displayed is written to a screen file and may be recalled with a read command.

FSMCLR	9	Clear the Session Manager Standard screen by setting the reference line to be the next input line, and the reference line offset to be zero.
FSMSSCR	11	Retrieves from the screen file the raw data stream from which a single logical screen image was composed. The logical screen number normally following the header element is not used and must be set to zero. Instead, the screen number is passed as four 256 base elements in the data stream following the header.
FSMPFMAP	12	Turns on and off the mapping of PFK's 13-24 to 1-12. This feature is of use with terminals whose righthand keypad PF keys are numbered 13-24. The element following the 24-element header is 1 to turn on mapping or 0 to turn off mapping. If *□ARBIN* is used, the result is the previous setting.
FSMFETF	14	As for *FSMGETS*, but applies to *□ARBIN/□ARBOUT* full screens.
FSMOEC	15	Sets the overstrike escape character (OEC). The element following the 24-element header is the *□AV* index of the character to be used, or zero to run off OEC. If *□ARBIN* is used, the result is the previous setting.

Following the header's command is the logical screen number. Its range is from *□AV*[0-255] for screen numbers 0-255. Element number 7 of the *FSMINQ* result is the number of logical screens available.

2) The Data Stream

Following the 24 byte header is the data stream, which is valid for write and read commands only. Refer to IBM manuals GA27-2749 (3270 Component Description) and GL27-6999 (3270 Programming) for a more detailed data stream description. The commands in the data stream perform screen writes and reads as well as assignment of cursor position and attribute characters. A writing data stream must begin with a write command followed by a write control character. After that, the data stream contains any combination of orders and data. Although a screen read can be done by transmitting a read command (*FSMRD* or *FSMRDT*) with an empty data stream, data can be transmitted simultaneously. In this way information can be displayed and a response solicited.

The data stream is determined to be all IBM codes or all SHARP Order codes by checking the value of the first byte. Below is a list of valid SHARP Order codes:

a) First Byte (the command)

Command	□*AV*	Meaning
SOWRT	13	Writes the following orders and data to the screen. This command adds to any existing screen image.
SOEWC	11	The screen is cleared and reset to its primary size before processing the rest of the data stream.
SOEWA	12	As for *SOEWC* except the alternate (larger) screen size is available for writing.

b) Second Byte

The data stream's second byte is a write control character (*WCC*). The *WCC* character controls, among other things, keyboard resetting and alarm sounding (for devices so equipped). A keyboard reset clears the input inhibited condition. The most common *WCC*'s are listed below but refer to the IBM manual for a complete breakdown of the *WCC* byte (each one resets the modified data tag bit).

Command	□*AV*	Meaning
WCR	3	Reset the keyboard after the write and do not sound the alarm.
WCN	1	Do not reset the keyboard and do not sound the alarm.
WCRA	7	Reset the keyboard and sound the alarm.
WCNA	5	Do not reset the keyboard but do sound the alarm.

c) The Remainder

The remainder of the data stream consists of SHARP Order (SO) codes and data. SHARP Orders are used to position, define and format data, erase parts of the screen or insert the cursor. Below is a list of SHARP Orders with a brief description of any bytes that must follow, then a descriptive paragraph of each order.

Order	⎕AV	Bytes	Meaning
SOALLE	0		The following data is all in IBM data stream format (EBCDIC).
SOE	1	⎕AV[256 256⊤*COUNT*]	The following encoded *COUNT* bytes are in IBM data stream format (EBCDIC).
SOALLZ	2		All of the following data is in Z-codes
SOZ	3	⎕AV[256 256⊤*COUNT*]	The following encoded *COUNT* bytes are Z-codes.
SOSBA	4	⎕AV[*ROW*,*COL*]	Set the buffer address to row *ROW* and column *COL* on the screen.
SOEUA	5	⎕AV[*ROW*,*COL*]	Erase all unprotected areas from the current buffer address to location *ROW* and *COL*.
SOSF	6	*ATTRIBUTE*	Start a screen field at the current buffer location and assign it attribute *ATTRIBUTE*
SOSA	7	*TYPE*,*VALUE*	Set attributes for characters to be displayed.
SOSFE	8	⎕AV[*COUNT*],(*COUNT*×2)⍴*TYPE*,*VALUE*	Start field extended. *COUNT* is the number of type/value pairs.
SORA	9	⎕AV[*ROW*,*COL*],*CHAR*	Repeatedly write character *CHAR* from the current buffer address to address *ROW*,*COL*
SOMF	10	⎕AV[*COUNT*],(*COUNT*×2)⍴*TYPE/VALUE*	Modify field. To the attribute character starting at the current buffer address, set the attributes

			described in the type/value pairs that follow.
SOAID	14	*AID,ROW,COL*	Attention identification. Inbound data stream information including *AID* and cursor *ROW*, *COL*umn at input termination.
SOLF	15	*ROW,COL,DEPTH,WIDTH*,0,0,0,*FLAGS,PAD,PREFIX ATTRIBUTE,SUFFIX ATTRIBUTE,SOZ,COUNT*	Logical field description. Includes start positions, dimensions, pad character, leading and trailing attributes. *ROW, COL, DEPTH, WIDTH* and *COUNT*, are 256 base, 2-byte counts.
SOIC	16	⎕*AV*[*ROW,COL*]	Insert cursor at the current buffer address.

SHARP Order Codes - Detailed Description

1) All EBCDIC (SOALLE)

This order says that all following data and orders are in EBCDIC. If some of the orders or data following are not in EBCDIC, use SOE (see below) instead.

2) EBCDIC (SOE)

A 2 byte count follows. That many characters are then considered to be in EBCDIC. SOALLE and SOE are used when transmitting an order which has no SHARP Order counterpart or non Z-code data. Counts for this order are 256 base and 2 bytes long, so encode them using ⎕*AV*[256 256⊤p*EBCDIC-DATA*].

3) All Z-codes (SOALLZ)

As for SOALLE, the entire data stream from this point is considered to be Z-code. Use this order when writing text to the screen which requires no conversion.

4) Z-codes (SOZ)

As for SOE, there is a 2 byte, 256 base count immediately afterward indicating how many Z-code characters follow.

5) **Set Buffer Address** (SOSBA)

This order looks for 2 bytes indicating the row and column to which the buffer address should be set. The row and column are direct indices into □*AV*, so for example, *SOSBA*,□*AV*[10 20] sets the buffer address at row 10, column 20. Use this order in cases such as starting a field (see SOSF and SOSFE) which begin at the current buffer address.

6) **Erase Unprotected to Address** (SOEUA)

This order is followed by a row and column address. All unprotected data is erased from the current buffer address to the address □*AV*[*ROW*,*COLUMN*].

7) **Start Field** (SOSF)

A field is started at the current buffer address. The field attribute character which must follow this order remains in effect until the next field attribute character is encountered. When the screen contains a single attribute character, the entire screen assumes the characteristics of that attribute. Attributes must be one of the following:

□*AV*	Meaning
76	alphanumeric input/output, non-display.
64	alphanumeric input/output, normal display.
200	alphanumeric input/output, high intensity.
92	numeric input/output, non-display.
80	numeric input/output, normal display.
216	numeric input/output, high intensity.
108	alphanumeric output, non-display.
96	alphanumeric output, normal display.
232	alphanumeric output, high intensity.
124	numeric output, non-display (note: any of these 3 cause cursor skip).
240	numeric output, normal display.
248	numeric output, high intensity.

Note: A field attribute character occupies physical space on the screen and appears as a blank character.

8) **Set Attribute** (SOSA)

This order is used to control extended features (on units with that option). Included in the extended features are color, extended highlighting and program symbols. Attributes set by the SOSA order remain in effect until another SOSA order is encountered, and override field attributes set in SOSFE orders. Subsequent data has those characteristics.

9) **Start Field Extended** (SOSFE)

SOSFE looks for a count byte (`⎕AV[COUNT]`) which indicates how many type/value pairs follow (again, refer to the IBM manual GA27-2749 for a complete breakdown of types and values). SOSFE differs from SOSF in that it is used to assign extended attributes as well as standard attributes starting at the current buffer address. SOSFE and SOSF control field attributes, while SOSA controls the attributes of single characters in the data stream.

10) **Repeat to Address** (SORA)

Writes the character following the specified buffer address from the current buffer address to the specified one.

11) **Modify Field** (SOMF)

Used to selectively change attribute characters previously set by SOSF or SOSFE at the current buffer address. The address must be that of an existing field attribute character and the data following it is identical to that of SOSFE.

12) **Erase Write Clear** (SOEWC)

The screen is cleared and reset to its primary size before processing the rest of the data stream.

13) **Erase Write Alternate** (SOEWA)

As for SOEWC except the alternate (larger) screen size is available for writing.

14) **Write** (SOWRT)

Writes the following orders and data to the screen. This command adds to any existing screen image.

15) **Attention Identification** (SOAID)

The next 3 bytes are an *AID* character and a 2 byte buffer address. From the AID character, an application knows how fullscreen input was caused. Termination codes include the enter key, clear key, a program function key or a PA key. A numeric value for the 2 byte address is achieved by *ROWCOL*←⎕*AV*⍳*ADDR*. Following the address is the data from any modified fields. SOAID is valid for an incoming data stream only (resulting from a SORD or SORDT).

AID	⎕*AV*
ENTER	125
PF1	241
PF2	242
PF3	243
PF4	244
PF5	245
PF6	246
PF7	247
PF8	248
PF9	249
PF10	122
PF11	123
PF12	124
PF13	193
PF14	194
PF15	195
PF16	196
PF17	197
PF18	198
PF19	199
PF20	200
PF21	201
PF22	74
PF23	75
PF24	76
PA1	108
PA2	110
PA3	107
CLEAR	109

16) **Logical Field** (SOLF)

Use the SOLF order to describe a logical screen field. A logical field has a defined area and attributes. SOLF is followed by 2-byte, 256 base *⎕AV* indices for starting row, starting column, depth, width, 3 reserved bytes, flags (explained later), a pad character, two properly formed SOSF, SOSFE, or SOMF orders, SOALLZ or a properly formed SOZ order, then any text to be written in that field. Even when data is not present, the SOZ order must be present with a length count of *⎕AV*[0 0]. Row, column, depth, and width are 256 base, 2-byte counts.

A suffix field is required since the prefix field attribute remains in effect until another field attribute is encountered. Flag bits are used to inhibit display of all or part of the SOLF data. This is desirable since the application may need to replace only an attribute (such as color) or just the data without affecting the rest of the field. The first 3 bits of the flag byte control writing of the prefix attribute, suffix attribute and data respectively. When a bit is on, that write is inhibited, so, to write data only, use *⎕AV*[2⊥1 1 0 0 0 0 0 0] as the flag byte (note: the trailing 5 bits are reserved). Any non-overstruck character may be used as a pad character, although the most common are null (*⎕AV*[254]) and blank (*⎕AV*[152]).

For example, to write *HI THERE* and attributes in a field whose start position is row 1 column 5 and whose depth is 10 and length 20, the SOLF order is:

```
SOLF,⎕AV[0 1,0 5,0 10,0 20, 0 0 0 0 152],
SOSF,ATTA,SOSF,ATTZ,SOALLZ,'HI THERE'
```

In the above example, *ATTA* and *ATTZ* are the prefix and suffix attribute characters as described in the IBM 3270 component description manual. Several SOLF order streams may be transmitted at once allowing several simultaneous screen field writes.

17) **Insert Cursor** (SOIC)

The cursor is placed at the current buffer address. For example, to place the cursor at row 10, column 20, use *SOSBA,⎕AV*[10 20],*SOIC*.

Transmitting a Complete Header and Data Stream

By constructing a data stream for the device and a controlling header, *⎕ARBIN* and *⎕ARBOUT* allow an application complete control over an IBM 3270 screen (see the sections on header and data stream formation above).

Transmission is one of the following:

```
R←⎕ARBIN (CS,22ρ⎕AV[0]),DS
```

where *CS* is a command and a screen number and *DS* is an IBM data stream, or

```
□ARBOUT (CS,22ρ□AV[0]),DS
```

□ARBIN differs from *□ARBOUT* in that it gives a result, which is necessary in the case of a screen read. Although either *□ARBIN* or *□ARBOUT* can be used in most cases, *□ARBOUT* is preferable in a fully debugged system as its response time is less. All data streams (incoming or outgoing) using *□ARBIN* have a 24 byte header (provided by the application in the outgoing and by the system incoming). The result of *□ARBIN* contains *□AV*[0] at position 4 if the operation was successful and an error code otherwise.

Workspace 5 *UTIL*3270 contains functions to format and transmit data streams and to decipher error codes.

Error	Meaning
0	All OK
1	Send access method failure
2	Receive access method failure
3	Output exceeds buffer size
4	Command illegal
5	Header too short
6	Reserved fields not 0
7	Read data exceeded buffer size
9	Invalid logical screen number in header
11	Invalid length
12	Invalid count in SHARP data stream
13	Translation of SHARP data stream too large for buffer
14	Undefined SHARP Order code in data stream
15	SHARP Order, reserved fields not 0
16	SHARP Order, logical field order invalid
17	Invalid logical screen number as argument
18	Attempt to retrieve stored logical screen failed.

Publication Code: 0346 8412 E2